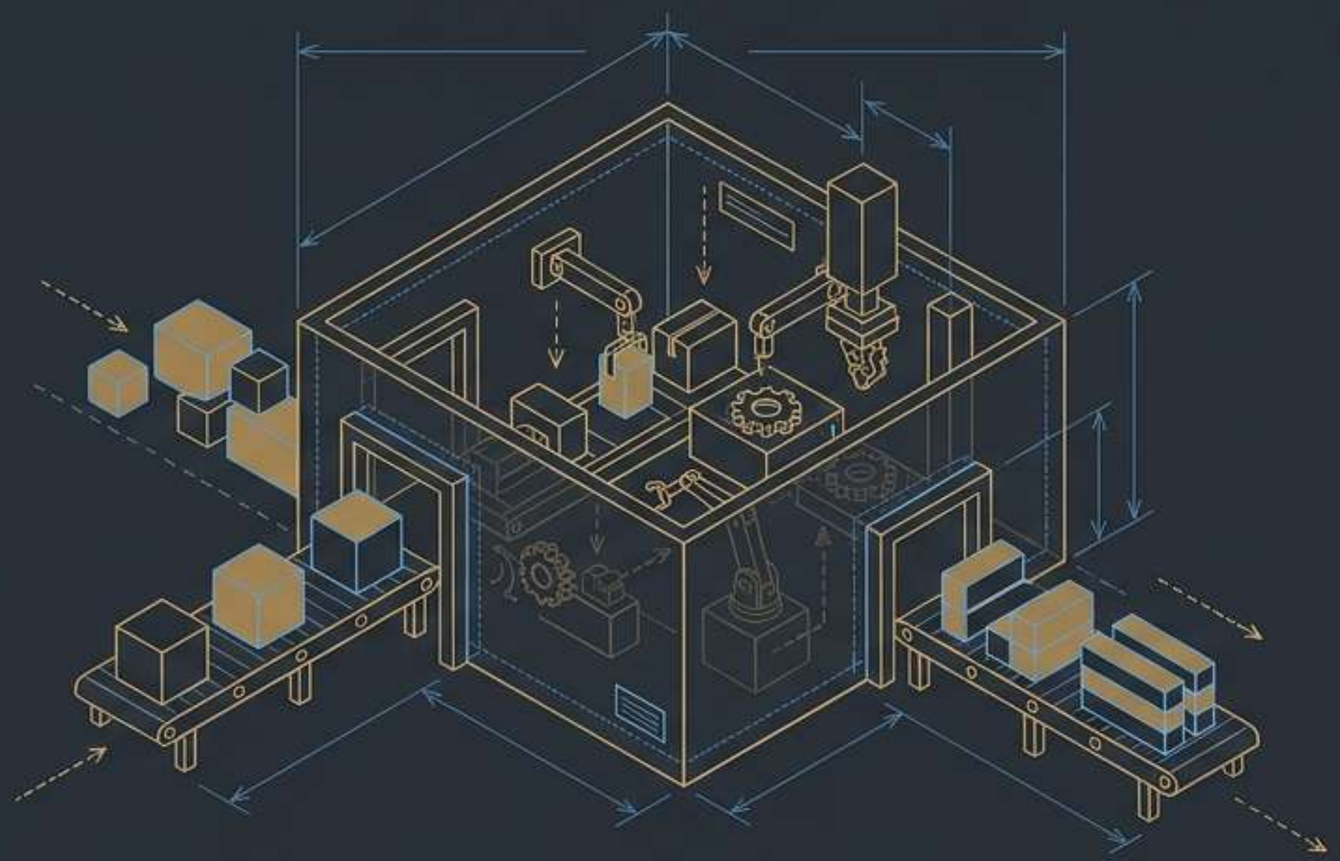
Fundamentos de Python

De texto plano a maquinaria funcional.

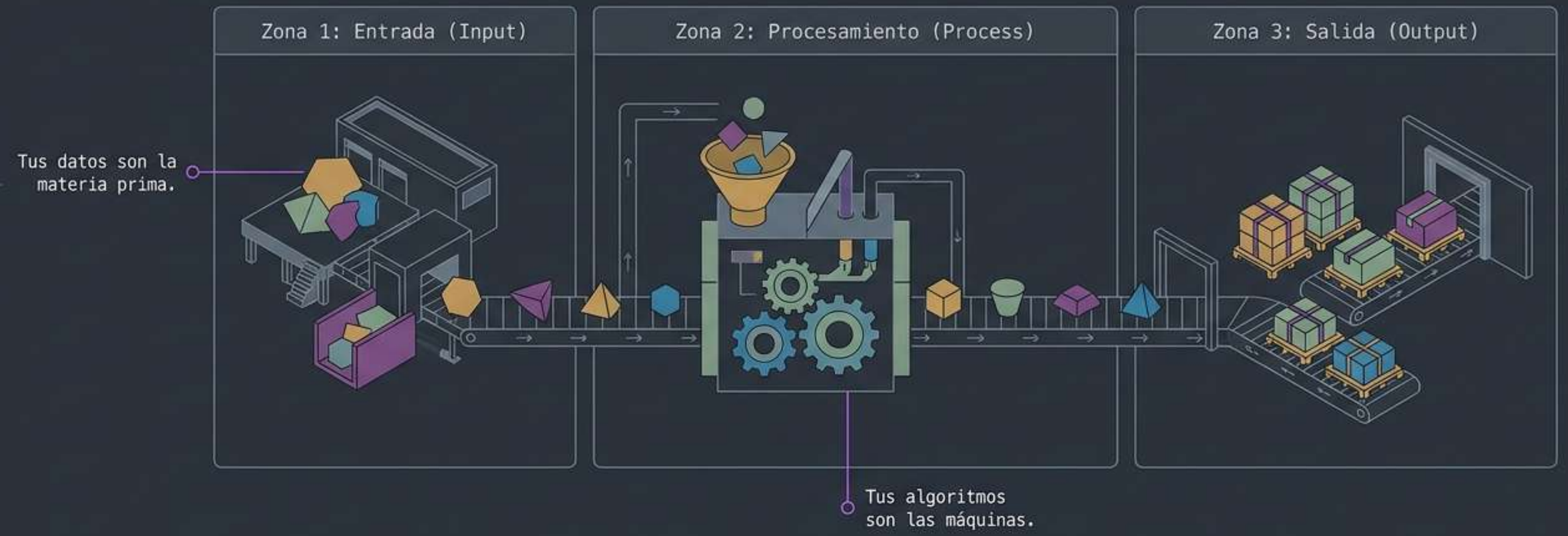


Construyendo con Código.

TIPO: BLUEPRINT_ARQUITECTÓNICO_V1

Pensar como un Arquitecto de Software

Programar no es simplemente escribir texto. Es diseñar una planta de ensamblaje automatizada. Cada línea de código es un conducto, una máquina o un sistema de control de calidad.



ID: PROCESO_FABRICACIÓN_SOFTWARE

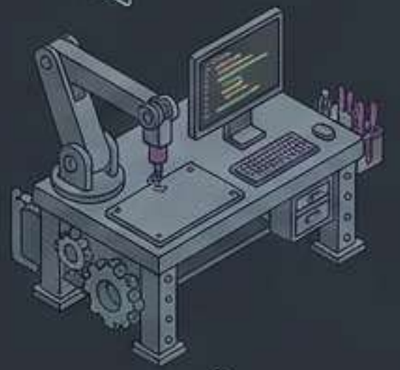
TIPO: DIAGRAMA_DECISIÓN_V1

Seleccionando tu Mesa de Trabajo

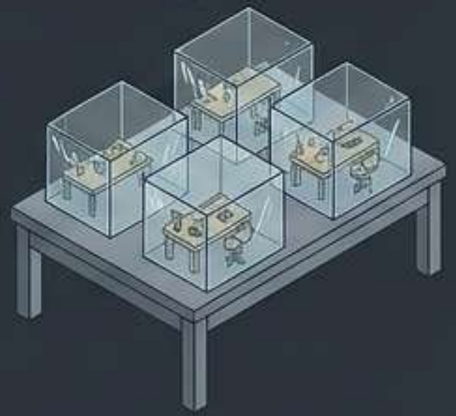
¿Qué necesitas construir hoy?

Exploración rápida sin instalación → Jupyter Notebooks / Google Colab

Proyectos escalables y control total → Instalación Local de Python



Múltiples proyectos sin conflictos → Entornos Virtuales



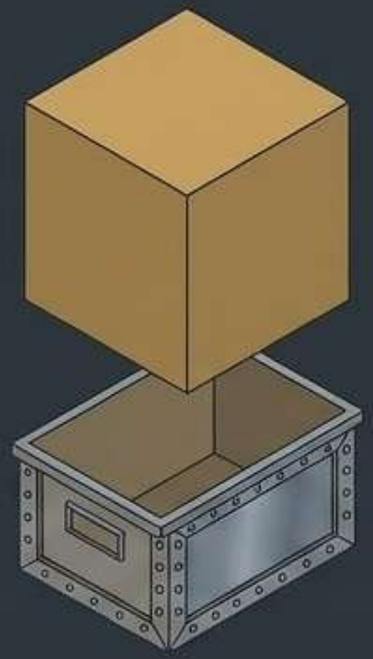
Los entornos virtuales actúan como laboratorios aislados; lo que ocurre en un proyecto no contamina al resto.

ID: SELECCIÓN_ENTORNO_TRABAJO

TIPO: MATERIA_PRIMA_VARIABLES_V1

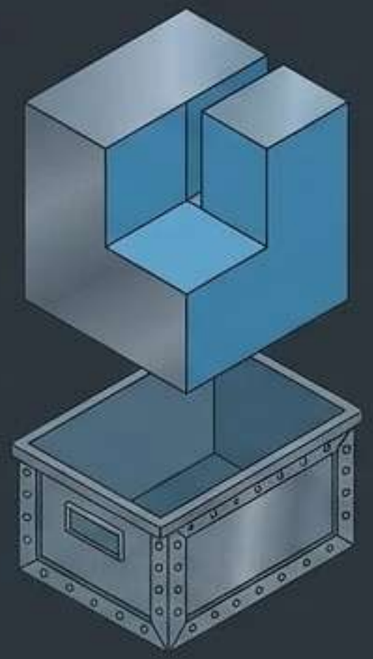
La Materia Prima del Código

Variables: Las variables son simplemente las cajas de almacenamiento etiquetadas donde guardamos estas piezas para usarlas más tarde.



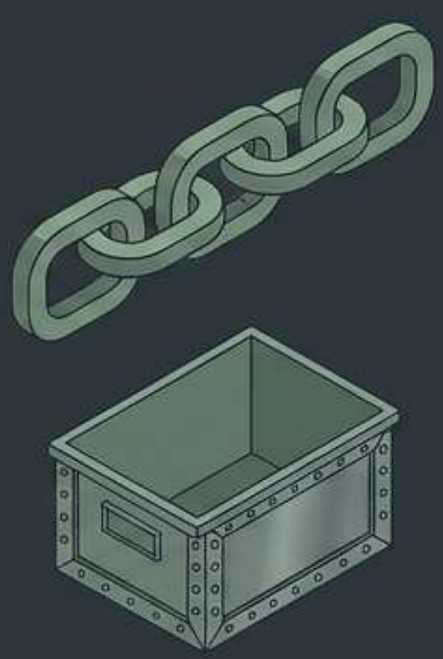
Enteros (Integers)

```
edad = 28
```



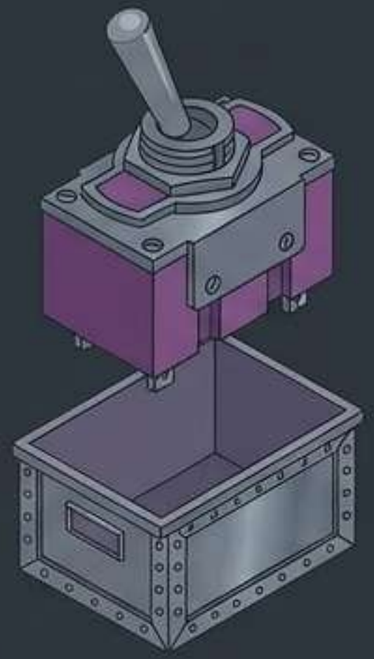
Flotantes (Floats)

```
precio = 19.99
```



Cadenas (Strings)

```
nombre = 'Ana'
```



Booleanos (Booleans)

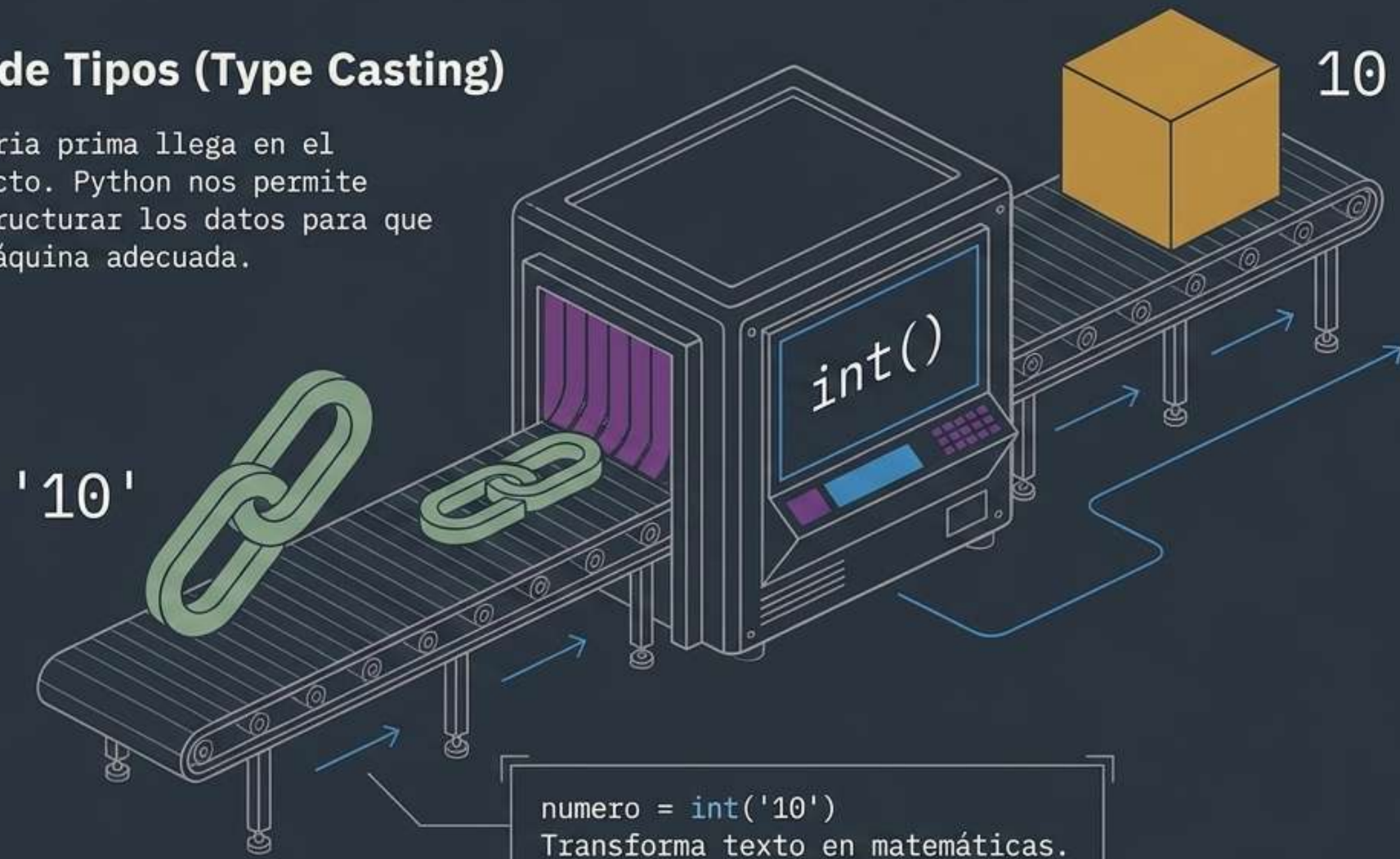
```
es_valido = True
```

ID: TIPOS_DATOS_BÁSICOS

Transformando Materiales

Conversión de Tipos (Type Casting)

A veces la materia prima llega en el formato incorrecto. Python nos permite 'fundir' y reestructurar los datos para que encajen en la máquina adecuada.

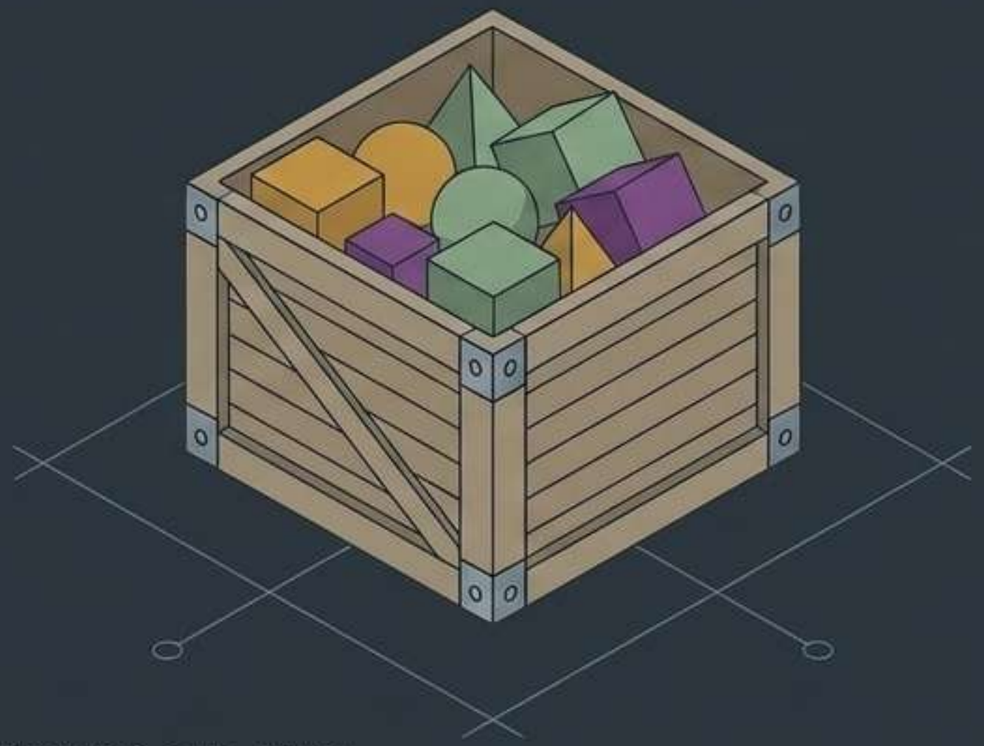



```
numero = int('10')
Transforma texto en matemáticas.
```

Contenedores y Logística

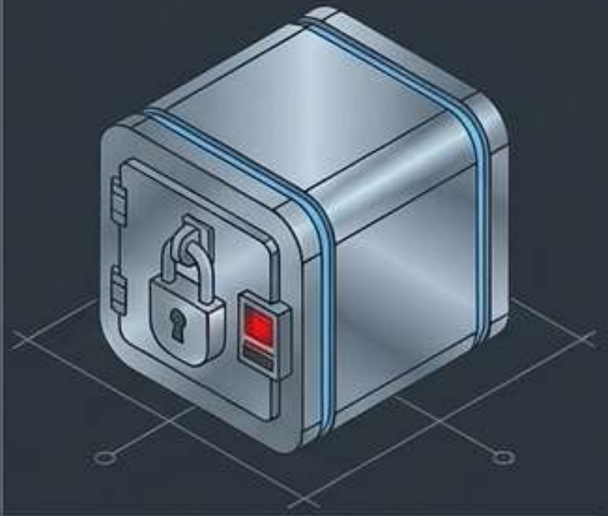
Las Estructuras de Datos Nativas de Python nos permiten agrupar la materia prima de manera eficiente. Elegir el contenedor correcto es el 80% del éxito en la programación.

Listas

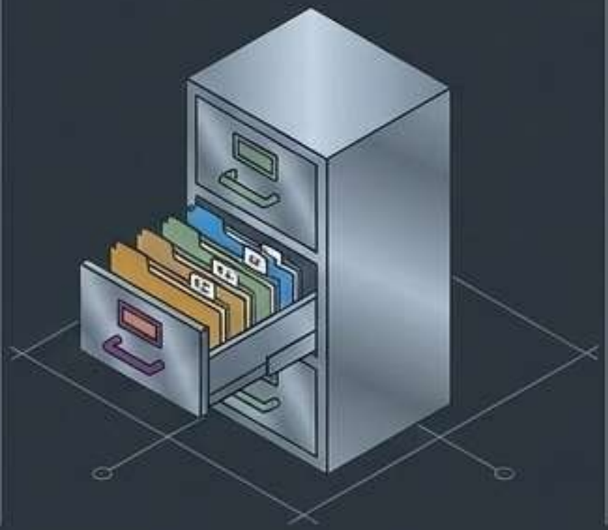


Listas

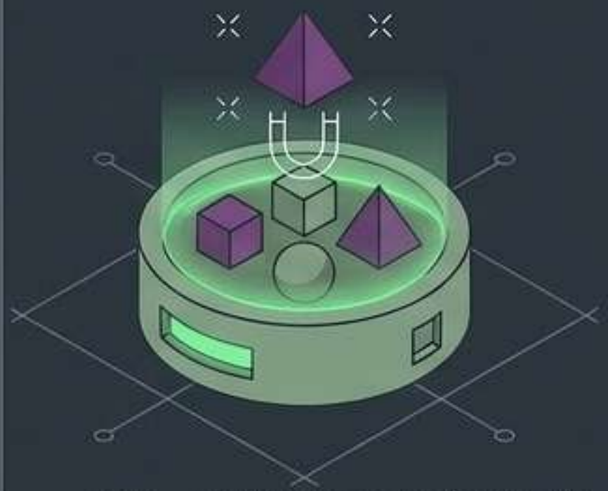
Tuplas



Diccionarios



Sets (Conjuntos)



ID: ESTRUCTURAS_DATOS_PYTHON

TIPO: CONTENEDORES_LOGÍSTICA_V1

El Catálogo de Contenedores

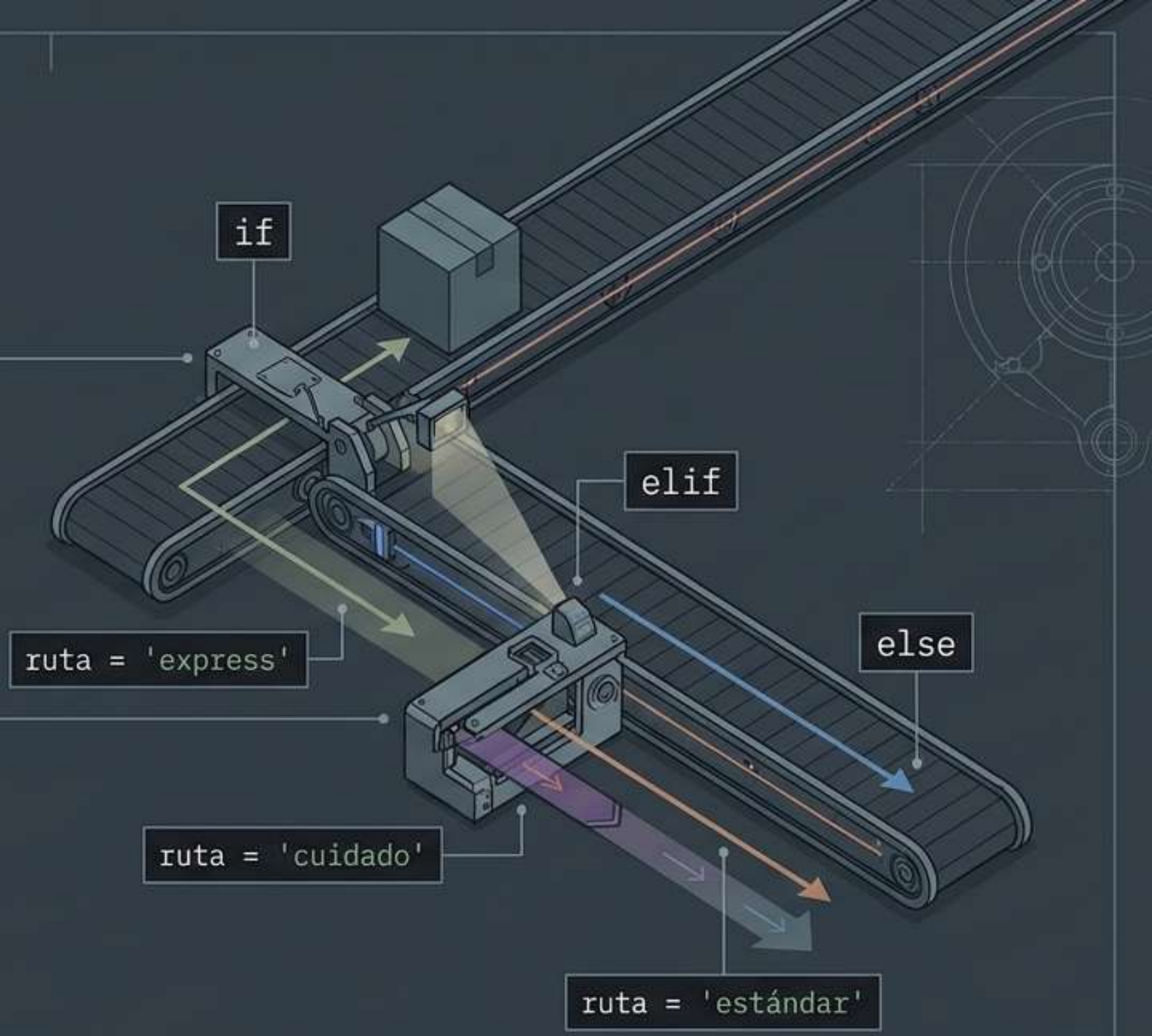
Estructura	¿Ordenada?	¿Mutable?	¿Admite Duplicados?	Uso Ideal
Listas []	Sí	Sí	Sí	Colecciones generales de datos que cambian con el tiempo.
Tuplas ()	Sí	No (Sellada)	Sí	Datos fijos que deben protegerse contra alteraciones (ej. coordenadas).
Sets {}	No	Sí	No	Filtrar elementos únicos o verificar pertenencia a alta velocidad.
Diccionarios {k:v}	Sí	Sí	No (en Claves)	Relacionar identificadores con información detallada.

Nota de Ingeniería: Usa Tuplas para seguridad, Sets para unicidad, Diccionarios para búsquedas rápidas, y Listas para todo lo demás.

Dirigiendo el Tráfico

Condicionales. Son los interruptores lógicos que deciden el destino de nuestros datos basándose en reglas estrictas.

```
if carga == 'urgente':
    ruta = 'express'
elif carga == 'frágil':
    ruta = 'cuidado'
else:
    ruta = 'estándar'
```

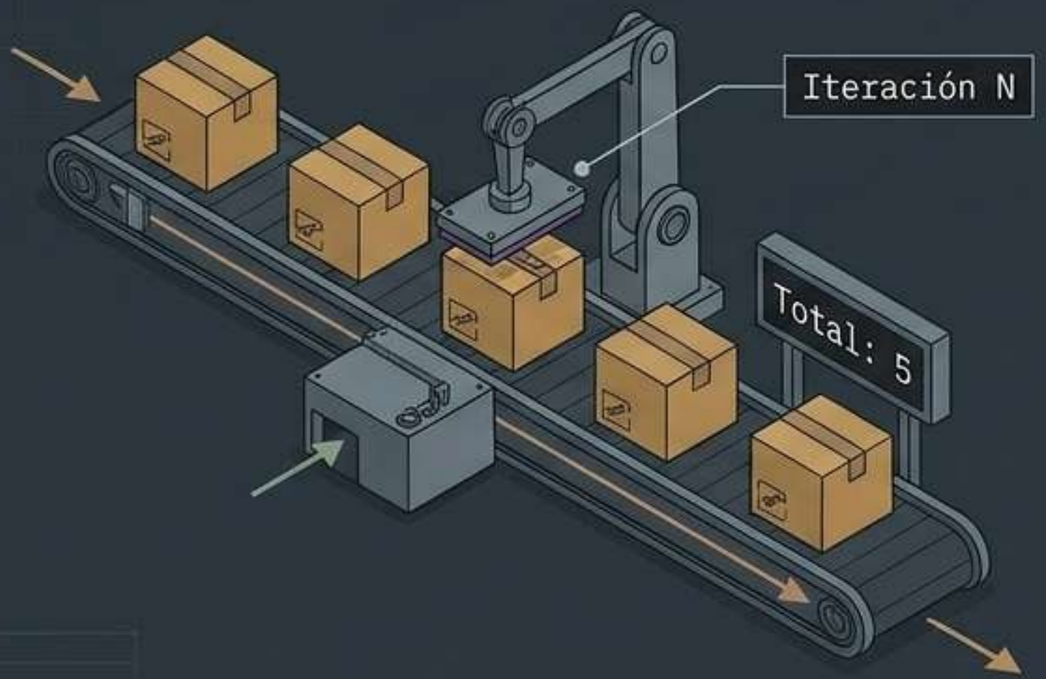


ID: CONDICIONALES_LOGICOS

TIPO: FLUJO_CONTROL_V1

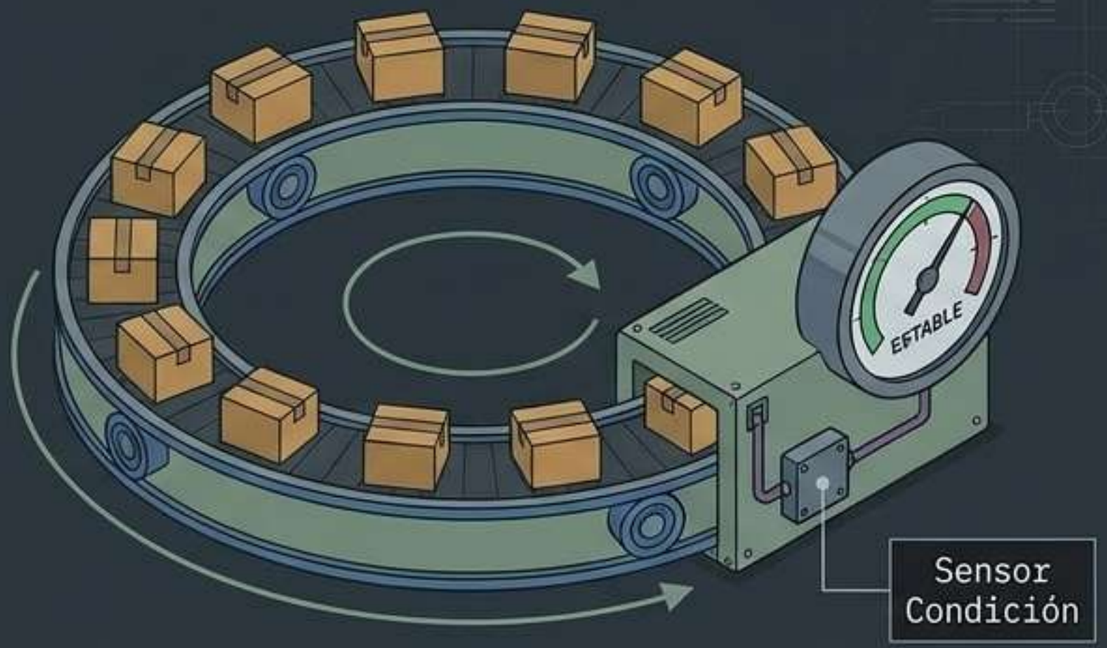
Automatizando la Cinta Transportadora

El Ciclo for



Iteración sobre una colección conocida. Se detiene automáticamente cuando se acaban los elementos.

El Ciclo while

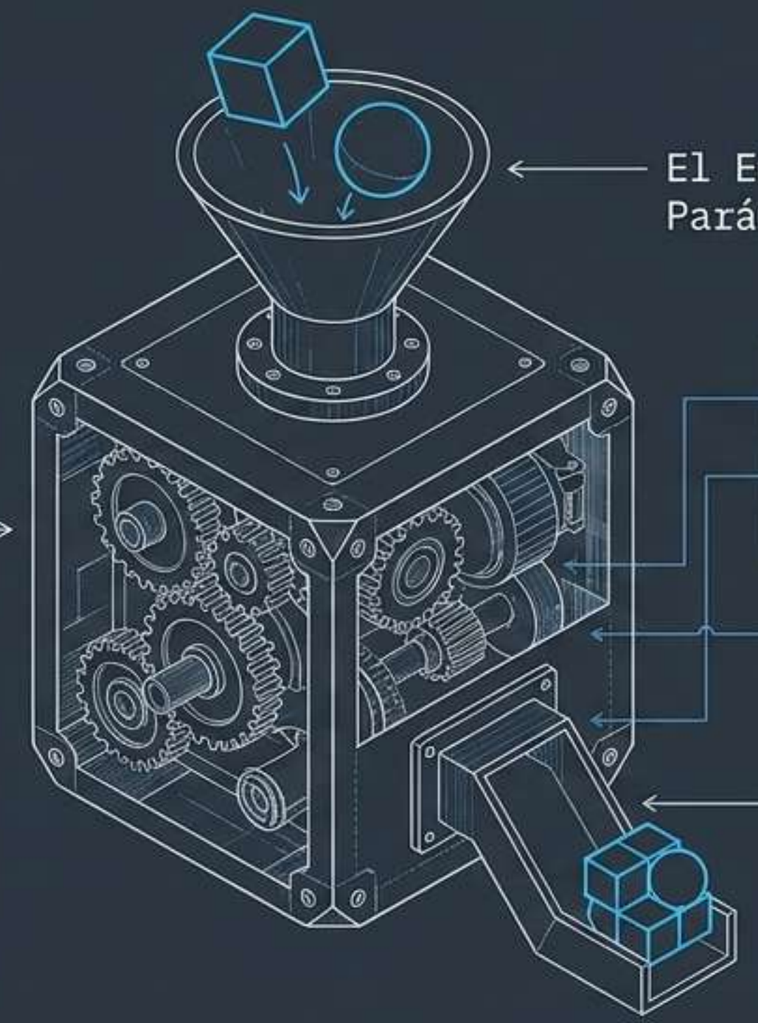


Ejecución condicional continua. Gira eternamente y se detiene solo cuando cambia el estado del sistema (el medidor baja).

Regla de oro: Usa `for` cuando sepas cuántas veces debes trabajar.
Usa `while` cuando dependas de una condición externa.

Construyendo tus Propias Máquinas

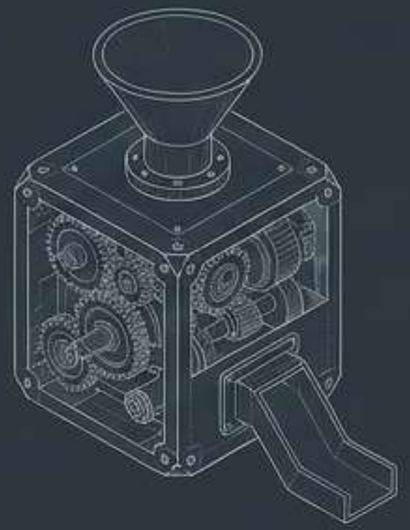
Las funciones encapsulan la lógica para que no tengas que reinventar la rueda. Construyes la máquina una vez y la usas mil veces.




```
def ensamblar_producto(pieza_a, pieza_b):
    resultado = pieza_a + pieza_b
    return resultado
```

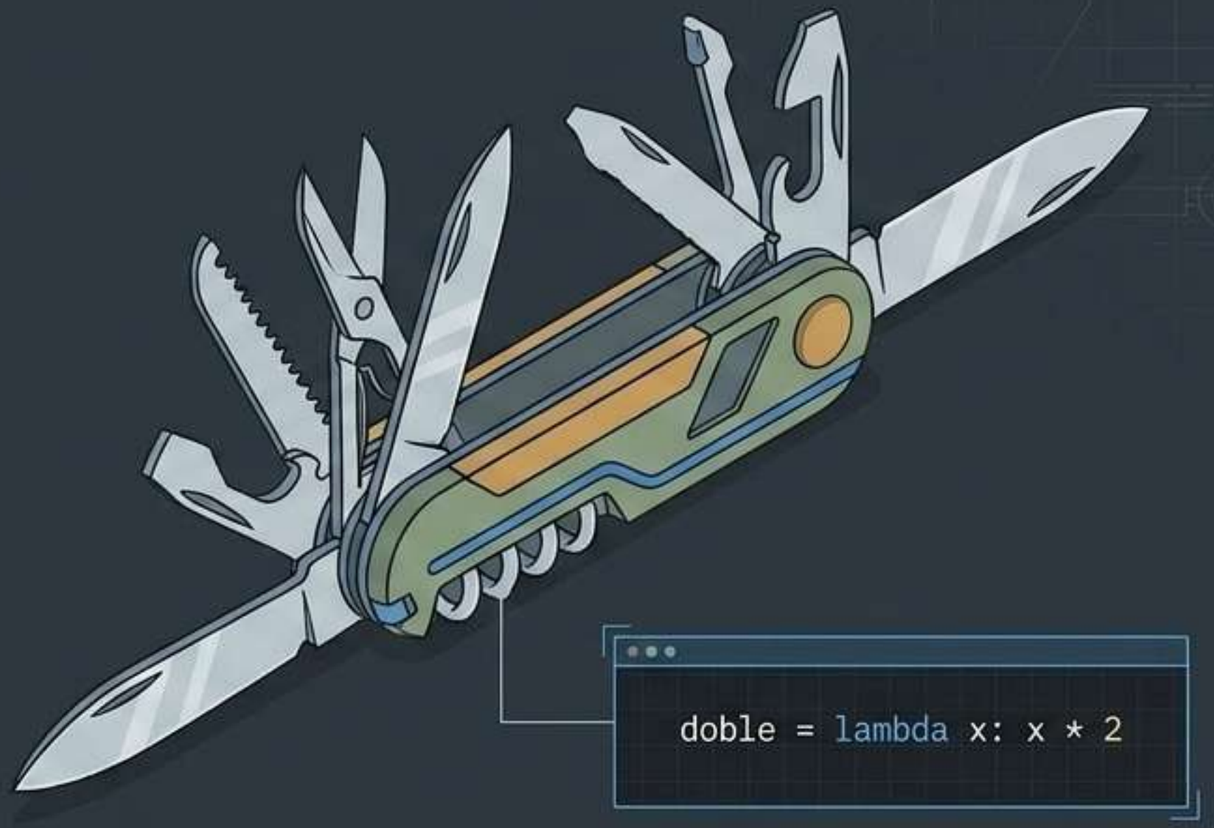
Herramientas de Bolsillo

Las Funciones Lambda son pequeñas operaciones de una sola línea. Perfectas para tareas desechables donde construir una máquina entera sería excesivo.



Función Estándar (def)

Robusta, multilínea, permanente.

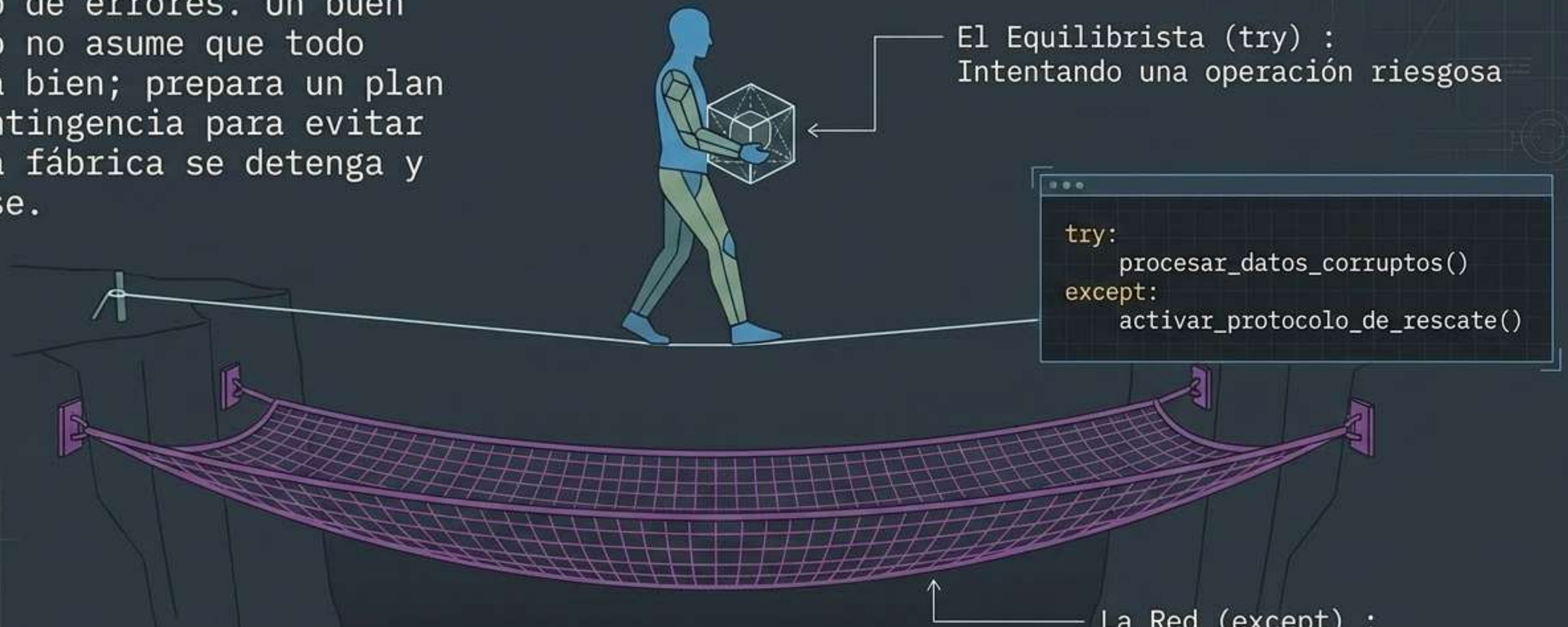


Función Lambda

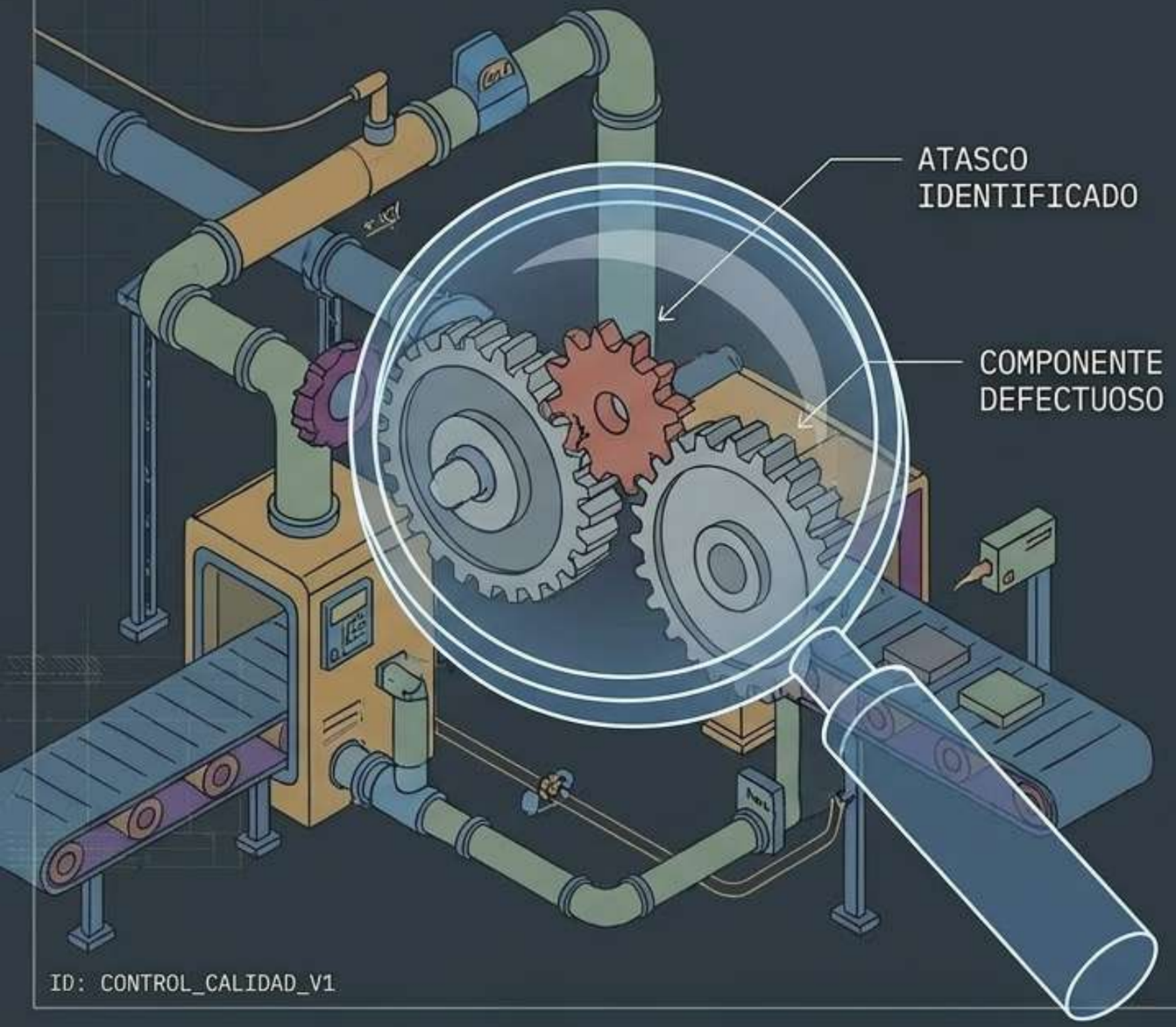
Rápida, anónima, de un solo uso.

La Red de Seguridad

Manejo de errores. Un buen código no asume que todo saldrá bien; prepara un plan de contingencia para evitar que la fábrica se detenga y colapse.



Control de Calidad en la Línea



1. Aislar el Componente

Usar la función `print()` para verificar el estado de los datos antes de que la máquina falle.

2. Leer el Registro (Traceback)

Python te dice exactamente en qué kilómetro de la cinta transportadora ocurrió el atasco. Nunca ignores el texto rojo.

3. Probar Hipótesis

Modifica una sola variable a la vez. Es el método científico aplicado directamente a tu código.

La Fábrica en Marcha

Todo programa complejo es simplemente la orquestación elegante de estas piezas fundamentales trabajando en perfecta armonía.

```
def procesar_inventario(lote):
    for articulo in lote:
        try:
            if articulo['estado'] == 'ok':
                print(articulo['id'])
        except KeyError:
            print('Dato faltante')
```

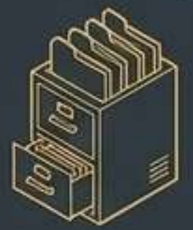
→ La Máquina (Función)

→ La Cinta Transportadora (Ciclo)

→ La Red de Seguridad

→ El Interruptor (Condicional)

→ El Contenedor (Diccionario)



El Siguiente Plano a Trazar

Conoces los materiales. Entiendes la logística. Tienes las máquinas.
La fábrica es tuya. ¿Qué vas a construir hoy?